APPENDIX A

Multi User NT KERNEL

1 OVERVIEW

The Multi User NT kernel sets up the operating environment in which the Winterm application programs execute.

2 KERNEL API

The kernel provides a variety of services to application and control programs. The services can be accessed through various interrupts:

```
Int FAh     Switch Between Virtual Mode and Protected Mode
Int FDh     Signal Session Switch
Int FEh / 0 Install Video
Int FEh / 1 Set Power-Down Time
Int FEh / 2 Get Video Selector
Int FEh / 3 Get Code Selector
Int FEh / 4 Get Data Selector
Int FEh / 5 Free Selector
Int FEh / 6
Int FEh / 7 Set Collection Number
Int FEh / 8 Disable Display Access
Int FEh / 9 Create Session
Int FEh / 10 Get Session Info
Int FEh / 11 Set Focus Session
Int FEh / 12 Destroy Session
Int FEh / 13 Mouse Event Handler
Int FEh / 14 New Graphics Mode
Int FEh / 15 Enable/Disable Session Switching
Int FEh / 16 Get Current Graphics Mode
Int FEh / 17 Prepare for Admin
Int FEh / 18 Acquire Resources
Int FEh / 19 Free Resources
Int FEh / 20 Post Thread Message
Int Feh / 21 Get Message
Int FFh / Boot Block Services
```

2.1 Int FAh: Switch Between Virtual Mode and Protected Mode

Interrupt FAh can be used to switch from virtual mode to protected mode, and then to return to virtual mode at a later time. After a transition, EIP will contain the offset of the next instruction after the int FAh instruction. Flags (except for the virtual mode bit) will be preserved across the transition.

Hardware interrupts are allowed while in protected mode.

Virtual mode service interrupts (i.e., DOS, BIOS, etc. (i.e., int 10h, int 14h, int 15h, int 16h, int 17h, int 21h, etc.)) should not be attempted while in protected mode.

When switching to protected mode:

> CS is set to the code selector which was created during the last invocation of int FEH, function 3. Normally this code selector is created to alias the current code segment.

Int FAh was defined to provide protected mode support for the Winterm/Citrix graphics driver, and is intended to be used with only a single code segment. Although it can be used with multiple code segments, the fact that there is currently no way to return to a previously created selector after a new selector is created, except by recreating the old selector, could be a limitation.

SS:ESP will be set to the kernel's stack. Operations that use the stack while in protected mode should assume a 32-bit stack. For example, ENTER, LEAVE, and other stack frame references should use EBP instead of BP.

All other segment registers will be set to arbitrary values, and should be initialized prior to use.

When switching to virtual mode:

A corresponding int FAh must have been executed previously to switch to protected mode.

All segment registers are restored to the values they had before entering protected mode. ESP is restored, as well.

2.2 *Int FDh: Signal Session Switch*

Switching between sessions is intended to be as transparent as possible to the applications running in the sessions that are being switched in and out of the foreground. To achieve this, the kernel maintains a virtual frame buffer for sessions when their applications are running in text mode. When text mode applications lose focus (i.e., they are switched out of the foreground), the contents of the physical text frame buffer is copied to the virtual buffer for that session. If the application continues to output text while it is in the background, the virtual buffer will be updated, and the application will have no knowledge that it is not writing directly to the screen. When that session is switched back to the foreground, the contents of the virtual buffer are copied back to the physical buffer.

Unfortunately, maintaining virtual screen buffers for graphics applications would require much larger amounts of memory, up to 768KB for a 1024x768 256-color mode. In order to maximize memory available for applications, the current architecture does not save virtual graphics screens. Instead, graphics-mode applications are required to save their own screen, if necessary, and to restore it when they are returned to the foreground. A proprietary interrupt is defined to signal graphics applications when the gaining or losing focus. (This is similar to the Win 32 paradigm, in which a REPAINT message is sent to an application to redraw part of all of its window.)

Interrupt FDh is issued by the kernel to a session that is running a graphics application that has focus when the kernel wants that session to give up its focus. The graphics application must issue its own interrupt FDh to acknowledge the kernel's request, and to signal that it is ready to give up the session.

Interrupt FDh is also issued by the kernel to a session running a graphics application when that session is about to receive focus. The graphics application should redraw the screen when it receives this signal.

2.3 *Int FEh / 0: Install Video*

```
Inputs:          AX = 0
Output:          None
```

2.4 *Int FEh / 1: Set Power-Down Time*

2.5 *Int FEh / 2: Get Video Selector*

```
Input:           AX = 2
```

```
Output:          AX = selector
```

The returned selector provides access to the graphics frame in protected mode.

The graphics chip is assumed to be in linear addressing mode.

2.6 *Int FEh / 3: Get Code Selector*

```
Inputs:          AX = 3
                 BX = code segment to alias
Output:          AX = code selector
```

This function creates and returns a selector that is an alias for a given code segment. The segment will have read and execute privilege when accessed through the new selector.

The selector should be returned with int FEh, function 5, when it is no longer needed.

2.7 *Int FEh / 4: Get Data Selector*

```
Inputs:          AX = 4
                 BX = data segment to alias
Output:          AX = data selector
```

This function creates and returns a selector that is an alias for a given data segment. The segment will have read and write privilege when accessed through the new selector.

The selector should be returned with int FEh, function 5, when it is no longer needed.

2.8 *Int FEh / 5: Free Selector*

```
Inputs:          AX = 5
                 BX = selector to return
Output:          None
```

This function returns a previously initialized selector to the kernel. The selector must have been initialized with int FEh function 3 or int FEh function 4.

2.9 *Int FEh / 6*

2.10 *Int FEh / 7: Set Collection Number*

```
Inputs:          AX = 7
                 BX = collection number
                 EDX = MS-DOS encoded file date and time
Output:          AX = previous collection number
```

2.11 Int FEh / 8: Disable Display Access

```
Inputs:          AX = 8
Output:          None
```

2.12 Int FEh / 9: Create Session

```
Inputs:     AX = 9
            EBX = session ID (0 if a new session is to be created)
            ECX = pointer to structure:
                  Offset 0 = size of structure in bytes (DWORD)
                  Offset 4 = requested size of base memory in bytes (DWORD)
                  Offset 8 = requested size of XMS memory in bytes (DWORD)
                  Offset 12 = pointer to a string describing the session (DWORD)
            EDX = flags
                Bit 0 = 1 iff the session is allowed to have focus (i.e., iff
                        it is allowed to run as the foreground task);
                Bit 1 = 1 iff the session is to be given CPU time to execute
                        non-interrupt code;
                Bit 2 = 0 if there is to be no change in the session's current
                        focus status;
                      = 1 if the session is to be given focus immediately;
            ESI = pointer to command to execute (zero-terminated string
                  containing command name),
                  or zero if no command is to be executed at the moment.
            EDI = pointer  to command tail, or zero if no command is to
                  be executed at the moment.
Outputs:    AL = return code of the last DOS program that exited from the
                 specified session;
            EBX = session ID (EBX is unchanged from the input value unless a
                  new session is created);
            EDX = flags;
                Bit 8 = 1 iff the session has the focus;
                Bit 9 = 1 iff the session is executing a DOS program.
```

If EBX is set to zero, a new session will be created (if there is enough memory). If EBX refers to an existing session, this function can be used to change that session's flags and description string, and/or to begin execution of a DOS application in that session. Currently, only the keyboard driver (KEYBD.COM) is executed automatically for each newly created session. (If the description string is not to be changed, either the pointer to the string or the pointer in ECX must be zero.)

COMMAND.COM is executed automatically when the first session is created at boot time; however, it is not executed automatically when this function is called subsequently to create additional sessions. COMMAND.COM can be executed by specifying it in a command line pointed to by the ESI register as described above, but this is normally done only for testing, because various application programs can be executed directly without invoking COMMAND.COM, and COMMAND.COM automatically executes AUTOEXEC.BAT, which should normally be executed only once, for the first session.

2.13 Int FEh / 10: Get Session Info

```
Inputs:     AX = 10
Output:     EBX = session ID of the current session;
            ECX = focused task;
            EDX = flags;
                Bit 0 = 1 iff the session is allowed to have focus (i.e., iff
                        it is allowed to run as the foreground task);
                Bit 1 = 1 iff the session is to be given CPU time to execute
```

```
        non-interrupt code;
Bit 8 = 1 iff the session has the focus;
Bit 9 = 1 iff the session is executing a DOS program;
Bit 10 = 1 iff a previously executed call to "Set Focus Session"
         Is still in progress.
```

Flag bit 9 can be polled to determine whether a program is still executing in the session, or whether it has exited. After an executing program terminates, bit 9 will be zero. Note that the session still exists, and will continue to exist until destroyed by int FEh function 12; however, no program is executing in the session. It is possible to begin execution of a new program by passing the name of the program through int FEh function 9.

2.14 Int FEh / 11: Set Focus Session

```
Inputs:          AX = 11
                 EBX = new task to which to give focus
Output:          None
```

This function specifies which application session will have focus. A session can only receive keyboard and mouse information if it has focus. If a session is running a text-based application, output from the application will go to the physical screen frame buffer only if the session has focus; otherwise, the output will go to a virtual buffer that is restored to the physical buffer when the session regains focus.

This function begins the session switching procedure and might return before the session switch completes. Function 10 can be used to verify that a session switch has completed.

Note that if a foreground graphics application uses this function to switch to the background, and then polls function 10 to see when the switch completes, the application must insure that interrupt FDh can be processed and acknowledged while the polling takes place.

The session that is losing focus must be allowed CPU time in order to execute the procedure to save the current graphics state. I.e., flag bit 1 (see int FEh function 9) must be set. The the CPU's time slice is disabled for that session, the "Set Focus Session" call will never complete.

2.15 Int FEh / 12: Destroy Session

```
Inputs:          AX = 12
                 EBX = task that is to be destroyed
Output:          None
```

This function cannot be used to destroy the foreground (focused) session.

2.16 Int FEh / 13: Mouse Event Handler

```
Inputs:          AX = 13
Output:          None
```

2.17 Int FEh / 14: New Graphics Mode

```
Inputs:          AX = 14
Output:          None
```

2.18 Int FEh / 15: Enable/Disable Session Switching

```
Inputs:         AX = 15
                EBX = 0 to enable session switching
                EBX = 1 to disable session switching
Output:         None
```

This function controls whether or not the user has the ability to switch sessions by pressing the session switch hot key.

Disabling session switching with this function does not prevent a program from switching sessions with int FEh function 11.

2.19 Int FEh / 16: Get Current Graphics Mode

```
Inputs:         AX = 16
Output:         AL = current graphics mode of the foreground session
                BL = bit field
                      Bit 0 = 1 iff there is a keyboard present
```

This function returns the current graphics mode of the foreground session. (This is included in SNMP information.)

2.20 Int FEh / 17: Prepare for Admin

```
Inputs:         AX = 17
Output:         None.
```

Sessions are closed and XMS memory is returned to the free pool in preparation for a firmware upgrade requiring all system memory to hold the new firmware image.

2.21 Int FEh / 18: Acquire Resources

```
Inputs:         AX = 18
                EBX = resource
                      0 - serial port 0
                      1 - serial port 1
                      2 - parallel port
                      3 - audio
                      4 - PC card 0
                      5 - PC card 1
                      6 - exclusion lock to perform terminal administration
                      7 - mass storage (flash memory or diskette drive)
                ECX = time-out interval in system timer ticks (18.2Hz) (zero
                      if there is no time-out)
                EDX = option bits
                      Bit 0 - set iff the kernel should display a dialog
                              box to confirm the transfer of ownership after
                              the time-out period expires
                      Bit 1 - set iff the interrupts generated by the
                              specified device should terminate power saving
                              mode (for serial ports that are used for
                              pointing devices)
                      Bits 2-15 - unused; set to zero
                SI = 0 (DS:SI might be defined later to point to a string
                     identifying the new owner of the resource, if the request
                     succeeds)
```

```
                DI = 0 (ES:DI might be defined later to point to a buffer
                     that will be filled with a string identifying the session
                     that currently owns the requested resource (if any) (81
                     chars max, including zero terminator)
                EBP = acquisition handle (zero for new requests)
Output:         Carry = set iff the resource could not be acquired
                EBX = id of the session owning the requested resource, if it
                      could not be acquired
                EBP = acquisition handle (zero if the request failed;
                      unchanged for requests to extend the timeout for
                      resources that have already been acquired; unique for
                      successful new requests)
```

This function allocates resources for exclusive use by the calling process. This will prevent improper operation caused by other processes trying to access the same device at the same time. All processes that need to use any of the devices covered by this function must call this function prior to using them. (Access to the devices will not be disabled by the kernel, but processes will not be guaranteed exclusive access unless all processes follow this protocol.)

Normally, resources that are acquired with this function must be returned when they are no longer needed, so that another process may use them. Function 19 is used to return acquired resources. If a session is destroyed, any resources that are owned by processes in that session will be automatically freed.

A time-out can be specified in the CX register. If an acquired resource's timeout expires and bit zero of DX is zero, that resource will be automatically freed. If an acquired resource's timeout expires and bit zero of DX is one, then that resource will still be owned by the process that acquired it, but it will be transferable if another process requests it. If another process attempts to acquire it, the kernel will display a message requesting confirmation that the resource should be transferred to the new process. If the user approves, the acquisition will be granted; otherwise, it will fail, and there will be no change in the resource's status. If an owning process frees a resource whose timeout has expired, the kernel will allocate that resource to the next requesting process without any user message.

In the initial implementation, the kernel will have no capability of displaying messages. If bit zero of DX is set, then a specified resource will be allocated with an infinite timeout.

If a process acquires a resource with a timeout, then the acquisition function must be called again, repeatedly, in order to extend the timeout period if the resource is actively used. For example, a process that is driving a printer might acquire the parallel port resource with a timeout period of five minutes, so that the parallel port can be reallocated to another process if it is idle for more than five minutes. However, if the original process continues to use the printer, it must continue to call the acquisition function (with a five minute timeout) (with EBP set to the acquistion handle that was returned during the original resource request) to reset the timeout period back to five minutes. The calling process should not call the acquisition function too frequently for performance reasons. The printing process, for example, would probably not want to call the acquisition function for every character.

Resources that are known by GUI.EXE to be required by an application for the entire duration of the application session's existence (e.g., for the application's main communication channel) should be allocated by GUI.EXE, rather than the application.

Resource six, "exclusion lock to perform terminal administration", should be acquired by any program performing local or remote administration (configuration or firmware updates) for the duration of the operation.

2.22 Int FEh / 19: Free Resources

```
Inputs:         AX = 19
                EBX = resource to free
                      0 - serial port 0
                      1 - serial port 1
```

```
                    2 - parallel port
                    3 - audio
                    4 - PC card 0
                    5 - PC card 1
                    6 - exclusion lock to perform terminal administration
              EBP = acquisition handle that was returned by function 18 when
                    the resource was acquired
Output:       None.
```

This function frees resources allocated by function 18. Specified resources that are not allocated, or that have already been freed, or that are allocated to a different process will not be affected.

2.23 Int FEh / 20: Post Thread Message

```
Inputs:       AX = 20
              EBX = session that is to receive the message
                    (zero for broadcast to all sessions)
              ECX = message ID
                        0-3FFh  system-defined messages
                     400h-7FFFh private window classes
                    8000h-BFFFh private messages
                    C000h-FFFFh messages registered by RegisterWindowMessage
              EDX = first message parameter
              ESI = second message parameter
Output:       None.
```

This function transmits a message to another session using the Win32 PostThreadMessage function.

** following message ids are used by Winterm2000 currently **

message id	meaning	1st message parameter	2nd message parameter
ABC0h	A permanent DHCP lease is obtained form DHCP server; new network configurations are stored in option.ini file	Network interface id (0=NI_TOKENRING; 1=NI_ETHERNET)	N/A
ABC1h	Request to shutdown the terminal	bit0 set if show warning message for user bit1 set if reboot right after shutdown	N/A
ABC2h	Initialize the COM port	0=COM1 1=COM2	N/A
ABC3h	Request to destroy all application sessions (i.e., all sessions except for the GUI session and the network session) in preparation for remote administration (sent by NETADMIN.EXE to GUI.EXE)	N/A	N/A
ABC4h	Notify GUI that an application session using TCP protocol has lost contact with the peer host due to network error (sent by TCPIP.EXE to GUI.EXE)	Session Id of the application session	Error code , specifying what type of network error
ABD0h	Acknowledgment for ABC3h (sent by GUI.EXE to NETADMIN.EXE after the request has been processed)	N/A	N/A
ABD1h	Acknowledgment for ABC4h (sent by GUI.EXE to TCPIP.EXE after the request has been processed, i.e. an application	Session Id of the application session being closed	N/A

session was closed due to the error reported
by message ABC4h)

2.24 Int FEh / 21: Get Message

```
Inputs:         AX = 21
Output:         Carry = set iff there is no message to retrieve
                ECX = message ID
                EDX = first message parameter
                ESI = second message parameter
```

This function checks the message queue, and, if there are any messages, retrieves the next message in the queue. If there are no messages, the function returns with the carry flag set.

2.25 Int FEh / 22: Get System Info

```
Inputs:         AX = 22
Output:         AL = currently selected keyboard language.
```

-

2.24 Int FFh: Boot Block Services

Interrupt FFh functions are passed on to the boot block's service handler. A function number is passed in the AX register. Functions are documented in the boot block documentation.

3 INTERRUPTS

Because the kernel runs in protected mode, all interrupts are received by the kernel. Depending on which interrupt is received, an interrupt will either be processed by the kernel or converted to a DOS interrupt that is issued in a virtual mode session that has a handler for a session.

It is possible to configure a session so that it will not receive a time slice from the CPU by clearing bit one of the session's creation flags. However, this will not affect interrupt processing within such a session. The session will still receive CPU time to process interrupts.

3.1 System Timer

The system timer interrupt occurs 18.2 times every second. When it occurs, the kernel issues a timer interrupt (interrupt 8) in every session.

Each session will have its own BIOS timer interrupt handler that will increment the tick count in the BIOS data area (virtual mode address 40h:6Ch). DOS applications are free to redirect the timer interrupt by writing to the virtual mode interrupt table, just as they normally would.

Because the interrupt controller is currently not virtualized (i.e., we do not intercept a virtual mode session's port access to the interrupt controller and emulate a separate virtual interrupt controller for each session), the BIOS timer interrupt handler in each session does not issue an end-of-interrupt command to the interrupt. The kernel issues the end-of-interrupt, so there will be only one end-of-interrupt per physical hardware interrupt.

3.2 Network Interrupt

The ethernet controller generates a hardware interrupt on IRQ 10, or interrupt 72h. This interrupt is redirected to the DOS session hosting the network driver.

3.3 Stack Interrupt

Applications issue interrupt 61h to request network stack operations. Interrupt 61h is redirected to the DOS session hosting the network stack.

Parameters to interrupt 61h functions are passed in the registers. Many of the functions require pointers to memory spaces within the calling application's memory address space. For example, when calling a function to read network information, an application will include a pointer to a buffer where the data will be written. This buffer must be accessible to both the calling program and the network stack, which might be in different DOS sessions. To facilitate this, the kernel maps the region of the calling program's memory address space containing the buffer into the network stack's address space in the D0000h-DFFFFh address range.

Although the network stack supports multiple simultaneous network connections, the network stack is not reentrant. The kernel guarantees that the stack's reentrancy requirement will not be violated.

3.4 Mouse Interrupt

The mouse requires one of the kernel's most complicated interrupt schemes. IRQ 12h (interrupt 74h) is the hardware interrupt which the kernel picks up and sends to the virtual mode PS/2 BIOS handler in the IO.SYS that is in the session that holds the mouse driver. IO.SYS calls the event handler in the mouse driver. The mouse driver calls the event handler in the kernel. (Because the standard mouse driver cannot "CALL" the kernel directly, it calls a vector in IO.SYS, and IO.SYS invokes the kernel's int FEh function 13.) The kernel's mouse event handler takes care of moving the mouse on the screen. (The virtual mode mouse driver is instructed to make its mouse "invisible".) Then the kernel invokes the event handler of the application running in the session that currently has focus (which might be different from the one containing the virtual mode mouse driver, and different from the one that was executing when the interrupt occurred).

(There are additional mechanisms to set up all of these vectors, in the right order.)

Although some of the PS/2 BIOS code will be duplicated, this scheme will make it possible to have only one mouse driver in memory, and the one driver will service all sessions. Furthermore, that one driver can be a standard mouse driver (i.e., the PS/2 mouse driver that we developed, the Microtouch driver, the Elo touch driver, or the light pen driver).

4 DEVICE VIRTUALIZATION

TABLE 1

OBJECT	DESCRIPTION
Wyse Management Info. Blocks (MIB)	
wbt2RamNum	the number of rows in the RAM table
wbt2RamIndex	the index number of the current table entry
wbt2RamType	1=base; 2=video; 3=extended
wbt2RamSize	the size of the RAM, in bytes
wbt2RomNum	the number of rows in the ROM table
wbt2RomIndex	the index number of the current table entry
wbt2RomType	1=boot; 2=OS
wbt2RamSize	the size of the ROM, in bytes
PCMCIA Devices	
wb2PCMCIANum	represents the number of rows in the table reporting inserted PCMCIA device(s)
wbt2PCMCIAIndex	the index number of the current table entry
wbt2PCMCIAType	0=a multifunction PCMCIA card; 1=memory; 2=modem; 3=parallel port; 4=fixed disk; 5=video adapter; 6=LAN adapter; 7=AIMS; 65535=empty
wbt2PCMCIAVendor	the string provided by the PCMCIA card vendor
IO Device Reporting	
wbt2IODevAttached	this object is a group of 32 flag bits
bit 0	keyboard attached
bit 1	mouse driver loaded
bit 2	MicroTouch screen driver loaded
bit 3	elo touch screen driver loaded
bit 4	light pen driver loaded
bit 5	floppy drive attached
Display Reporting	
wbt2DispFreq	refresh rate currently in use on the display, in Hz
wbt2DispHorizPix	number of pixels in a horizontal line of video currently in use on the Winterm
wbt2DispVertPix	number of lines of video currently in use on the display
wbt2DispColor	color depth currently in use on the display
wbt2DispFreqMax	maximum refresh rate available on the display

wbt2DispHorizPixMax	maximum number of lines of video available on the display
wbt2DispVertPixMax	maximum number of lines of video available on the display
wbt2DispColorMax	maximum color depth available on the display
DHCP Info. Reporting	
wbt2DhcpInfoIndex	index of the table entries with a range between 1 and the value of the wbt2DhcpInfoNum
wbt2InterfaceNum	number of the interface used to obtain the information in this table entry
wbt2ServerIP	name or IP address of the server which will be used for ICA connections which use this option and do not configure an explicit server
wbt2UserName	user name which will be used to log in to the server for ICA connections which use this option and do not configure an explicit user name
wbt2Domain	Windows domain which will be used to log in to the server for ICA connections which use this option and do not configure an explicit domain
wbt2Password	indicates the presence or absence of a password which will be used to log in to the server for ICA connections which use this option and do not configure an explicit password. 0=No password; 1=Password present
wbt2CommandLine	command line which will be issued once the user is logged in to the server for ICA connections which use this option and do not configure an explicit command line
wbt2WorkingDir	working directory which will be set once the user is logged in to the server for ICA connections which use this option and do not configure an explicit working directory
wbt2FileServer	name or IP address of an ftp server which contains Winterm base and application images for downloading and the PARAM.INI and PARAM#. INI files which allow the Winterm to determine if the currently loaded image is or is not current
wbt2FileRootPath	path to be used for obtaining images to be downloaded
wbt2TrapServerList	list of SNMP management servers which are to be notified when a trap event occurs
wbt2SetCommunity	value of the Set Community for this Winterm
ROM Image Info. Reporting	
wbt2CurInfoNum	this object represents the number of entries in the current ROM image information table
wbt2CurInfoIndex	index of the table entries with a range between 1 and the value of the wbt2CurInfoNum
wbt2CurInfoNum	
wbt2CurBuildNum	build number of the image currently loaded on the terminal
wbt2CurOEMBuildNum	OEM build number of the image currently loaded on the terminal

wbt2ModBuildDate	date the image currently loaded on the terminal was last modified
wbt2CurOEM	OEM whose image is loaded on the terminal
wbt2CurUser	user currently logged in to the UI on the terminal
wbt2CurHWPlatform	hardware platform installed on the terminal
wbt2CurOS	operating system installed on the terminal
wbt2CurRefreshRate	the display refresh rate supported by the image on the terminal
wbt2CurConfig	string describing the function of the image currently loaded on the terminal
wbt2DUpInfoNum	this object represents the number of entries in the update ROM image information table
wbt2DUInfoIndex	this is an index of the table entries with a range between 1 and the value of the wbt2DUpInfoNum
wbt2DUpBuildNum	build number of the image updated image available on the server
wbt2DUpOEMBuildNum	OEM build number of the updated image available on the server
wbt2DupModBuildDate	date the image on the server was last updated
Custom field content reporting	
wbt2CustomField1	reports the contents of Custom Field 1
wbt2CustomField2	reports the contents of Custom Field 2
wbt2CustomField3	reports the contents of Custom Field 3
New MIB Group	
wbt2UpDnLoadNum	this object represents the number of entries in the upload or download request table
wbt2UpDnLoadIndex	index of the table entries with a range between 1 and the value of wbt2UpDnLoadNum
wbt2upDnLoadId	unique ID string for this upload/download request
wbt2UpDnLoadOp	0=an upload request; 1=a download request
wbt2UpLoadSrcFile	path and file name for source file to be uploaded/downloaded
wbt2UpDnLoadDstFile	path and file name where the uploaded/downloaded file will be stored
wbt2UpDnLoadFileType	0=a binary file; 1=an ASCII file
wbt2UpDnLoadProtocol	0=FTP
wbt2UpDnLoadFileServer	file server IP (or name) to/from which the file is to be uploaded/downloaded
wbt2UpDnLoadTimeFlag	0=execute request immediately after wbt2SubmitLoadJob is set to ready (1)

wbt2AcceptReq(2):	"0"=no; 1=yes"
wbt2SubmitLoadJob(3):	"0=not ready; 1=ready"
SNMP Traps	
wbt2TrapDHCPBuild Mismatch	software information mismatch supplied by PARAMS.INI or PARAMS#.INI variables has been detected. A DHCP/FTP update will be attempted.
wbt2TrapDHCPUpdDone	A DHCP/FTP software download has been completed
wbt2TrapDHCPUpdNot Complete	A DHCP/FTP software download did not complete due to an error. The failure reason will be sent with this trap.
wbt2TrapSNMPAccptLd	Winterm is ready to accept an upload or download request initiated by the SNMP manager
wbt2TrapSNMPLdDone	SNMP initiated upload or download request has been completed. The wbt2UpDndLoadId will be sent with this trap.
wbt2TrapSNMPLdNot Complete	SNMP initiated upload or download request did not complete due to an error
wbt2UpDnLoadId	failure reason will be sent with this trap
TRAP Failure Reasons	
wbt2TrapsInfo(9)	
wbt2TrapStatus(1)	Current trap status information
LS_NOTREADY	The WBT is not ready to process this request
LS_FAIL_NOUPD	The enable SNMP/DHCP update field in the user interface has not been selected
LS_FAIL_ FILENOTFOUND	The source file requested cannot be located
LS_FAIL_UPLD_ BLOCKED	The upload process request is blocked by the WBT
LS_FAIL+_NOSERV	Cannot locate or connect to the specified server
LS_FAIL_PROT	The specified file transfer protocol has detected an error during a file upload or download
Set PDU function	
Set operation	limited to a specific community name definable in UI or by option 164 from the DHCP server
default set community name	WBTADMIN

SNMP Upgrade	
Winterm	powers up and sends the wbt2TrapSNMPAccptLd Trap
automated downloading	through OpenView or other SNMP management programs the administration program will be able to detect the wbt2TrapSNMPAccptLd Trap and through scripting identify the current revision of software and initiate file uploads or downloads to the Winterm 2000
SNMP	allows the current Winterm configuration files (.0NI) to be obtained via a FTP connection
Bundled image updates	handled identically as in DHCP image updates with the exceptions that:
FPT server, path and filenames	are specified via setting appropriate objects in the wbt2UpDnLoad group
download	is not conditional on the contents of a PARAM#.INI file
Winterm 2000	once the image update has been completed will automatically reboot
USER INTERFACE ENHANCEMENTS	
DEFAULT NETWORK CONNECTION	is changed to enable DHCP configuration
default SERVER NAME (132.237.250.50)	found in previous version of software is eliminated
LOAD NETWORK DRIVERS ON STARTUP	selection is changed from previous version to LOAD ETHERNET DRIVERS
Non APEX PCMCIA modems	using user defined strings are supported
network shutdown button	is added to the main connection menu of the user interface
Keyboard Languages	are displayed differently (asterisk, etc.) in the Winterm's Keyboard Language field list
NSMP/Network Administration screen	will be created
SNMP Communication	
Enable Authentication Failure Trap	toggle
Community	60 character string
Set Community	60 character string
Trap Server 1	60 character string
Trap Server 2	60 character string
Trap Server 3	60 character string

Trap Server 4	60 character string
Update of Firmware	
DHCP automated updates	enabled
SNMP updates	enabled
Terminal Location	60 character string
Contact	60 character string
User Directory Path	60 character string
Custom Field 1	60 character string
Custom Field 2	60 character string
Custom Field 3	60 character string
REMOTE ADMIN. SOFTWARE ENHANCEMENTS	
WyseWorks Remote Admin. tool	is expanded to provide additional functions
Binaries	WyseWorks tool is expanded to give the user the ability to create binaries with customized configurations from the toolkits
Citrix WinFrame/Microsoft WTS Installation Protection	restrictions are removed from the product so that it functions on Standard NT 3.51 or 4.0 workstations or servers
Command Line Options	allows automated configuration file and image (workspace) merges
Command Syntax	WWADMIN /I <input files path> /O <output files path> /W <workspace name (no extension)>
Exit Error codes returned	i.e., IF ERRORLEVEL type processing in BAT file
Unknown option	i.e. NOT /I /O /W
Unexpected option	i.e. /I /O (/I not followed by path)
Too many Too few	arguments
In path specified	not found
Out path specified	not found
Workspace path specified	not found
Failure to open Workspace definition file	(path exists)
Internal error	call a certified technician

TABLE 2

INI	ENTRY	DESCRIPTION	FORMAT	UPGRADE
BuildVer	PARAMS.INI displayed in the main "about" box PARAM#.INI not displayed	this string indicates the major Wyse Build Number	an 8 character string (#########) where the first 4 characters represent the major version number and where the last four characters represent the build number	used to determine if an image upgrade is required
OemBuildVer	PARAMS.INI displayed in the main about box PARAM#.INI not displayed	this string indicates the minor OEM Build Number	a 4 character numeric string (#####) where the four characters represent the OEM build number	used to determine if an image upgrade is required
ModDate	data not displayed in the "about" box	this string indicates the date and time the files were last modified	encoded as a 14 character numeric string (##################).	used to determine if an image upgrade is required
ImageFileName		this string indicates the filename of the image binary embedded in the terminal or in the current directory of the FTP server	an 8.3 character string with an ISO 9660 format (A-Z, 0-9, and underscore only)	used to determine if an image upgrade is required
OemName	data not displayed in the "about" box	this string indicates an OEM version of the software	an 8 character string with an ISO 9660 format (A-Z, 0-9, and underscore only	affects the image upgrade path on the FTP server

Option	always blank in PARAMS.INI and always defined in PARAM#.INI	this string indicates an option version of the software	an 8 character string with an ISO 9660 format (A-Z, 0-9, and underscore only	affects the image upgrade path on the FTP server
UserPath	data not displayed in the "about" box	this string indicates a custom user configured version path	a user interface field up to 60 characters	affects the image upgrade path on the FTP server
Platform	data not displayed in the "about" box	this string up to eight characters indicates the hardware version as WINDOWS, CLASSIC, or LEO	an 8 character string with an ISO 9660 format (A-Z, 0-9, and underscore only	affects the image upgrade path on the FTP server
OS	data not displayed in the "about" box	this string up to eight characters indicates the OS version as BOSS, CE, or LINUX	an 8 character string with an ISO 9660 format (A-Z, 0-9, and underscore only	affects the image upgrade path on the FTP server
Refresh	data not displayed in the "about" box	this string up to eight characters indicates the refresh version of the software sd 75HZ 60HZ, or FP	an 8 character string with an ISO 9660 format (A-Z, 0-9, and underscore only	affects the image upgrade path on the FTP server
WFCLIENT .INI				
version	data is displayed in the "about" box	this string definition remains the same as in previous versions	generally #.##C for Wyse Products	does not affect the image upgrade process
ACCT.INI				
FtpUser	data not displayed in the "about" box	this string indicates the user name to be used by the client during a FTP connection		this data affects the FTP connection process
FtpPassword	data not displayed in the "about" box	this string indicates password used by the client during FTP connection		this data affects the FTP connection process